XProcRef

Internal documentation

Erik Siegel - Xatapult
2024-09-05

0 Table of Contents

1 Introduction

XProcRef

Figure 1-1 - The XProcRef logo

XprocRef is an initiative to publish a website with descriptions for the XProc (3.0 and later) steps in a more user-friendly way than in the official specification. It is an initiative of Erik Siegel (Xatapult Content Engineering).

There is an underlying reason for this. In 2020 I published a book called “XProc 3.0 - Programmer Reference” (for sale here):

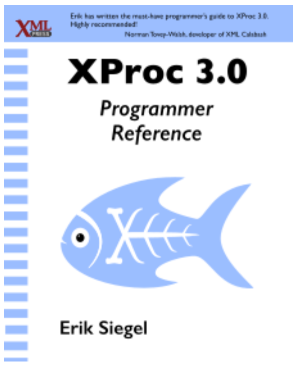
XML PRESS
Erik has written the must-have programmer's guide to XProc 3.0. Highly recommended!
Norman Tovey-Walsh, developer of XML Calabash
XProc 3.0
Programmer Reference
Erik Siegel

Figure 1-2 - The XProc Programmer reference book cover

Appendices A and B in the book describe the step library. However, due to time constraints, the step descriptions were copied from the formal XProc specification. This leaves much to be desired for users of the language: the specification is aimed at XProc processor *implementers*, not at language *users*. To correct this, this site contains reference information about the XProc steps, written from a more user-oriented perspective. With the increasing popularity of XProc, I hope this fills a need.

2 Technical overview

The website

- XProcRef (https://xprocref.org) is a static website containing descriptions of XProc steps. It is hosted as a GitHub Pages site from its GitHub repository https://github.com/xatapult/xprocref.
- It uses Bootstrap 5.2.2
- All static website resources (CSS, JavaScript, etc.) can be found underneath `xprocref/web-resources/`. This all is copied verbatim to the final website.
- The template used to produce the actual web pages is: `xprocref/web-templates/default-template.html`.

The markup language

- All is generated from an XML source: `xprocref/src/xprocref.src.main.xml`
- This source uses an XProcRef specific markup language that is, more or less, described in this document.
  - A W3C XML Schema for the markup language is in: `xprocref/xsd/xprocref.xsd`
  - A Schematron Schema for the markup language with additional validations is in: `xprocref/sch/xprocref.sch`
- For the content/text parts of the markup, DocBook 5 is used. XprocRef adds several extension elements to provide for things like step cross-references, special markup, etc.

Processing the markup into the website

- This markup language is processed into a static website by an XProc 3.0 pipeline: `xprocref/xpl3/process-xprocref.xpl`.

  > The pipelines are currently (2024-09-05) processed by the MorganaXproc-III EE (Enterprise Edition) processor. The SE (Standard Edition) processor will not do the job because some steps are used that are available in the EE version only.

- Processing makes heavy use of the facilities provided some other open source components published by Xatapult Content Engineering:
  - Xtpxlib Common
  - Xtpxlib Container
  - Xtpxlib Xdoc
- To ease the processing, there are some other pipelines that drive the main `process-xprocref.xpl` with the right set of options and port connections:
  - To produce a *test* version of the website (with all steps, also the unpublished ones): `xprocref/xpl3/process-xprocref-test.xpl`

    The test version is build in `xprocref/build` (which Git will ignore).
  - To produce a *production* version of the website (only the published steps): `xprocref/xpl3/process-xprocref-production.xpl`

    The production version is build in `xprocref/docs` (the default directory used for the GitHub Pages site).

    > The production pipeline currently (2024-09-05) sets an option `wip` to `true`, causing a “work in progress” warning on every page. Turn this off when the website is complete (if ever…).

Testing/Playing

- Facilities to play around with the markup language and for testing features are in `xprocref/test`.

3 The XProcRef markup language

3.1 Overall structure

This is not documented (yet). Please refer to the schema in `xprocref/xsd/xprocref.xsd`.

3.2 Describing a step

This is not documented (yet). Please refer to the schema (`<step>` element) in `xprocref/xsd/xprocref.xsd`.

3.3 XprocRef DocBook sections

DocBook sections are *not* validated (yet)!

For any element in the XprocRef markup that can contain text, the following applies:

- The basic language is DocBook 5 (the xtpxlib-xdoc dialect), with xtpxlib-xdoc extensions. These extensions can be used to, for instance, create descriptions of XML elements.
- Elements do *not* have to be in the DocBook namespace. Any element that is not in the DocBook 5 (`http://docbook.org/ns/docbook`) or xtpxlib-xdoc (`http://www.xtpxlib.nl/ns/xdoc`) namespace is automatically changed into the DOcBook namespace.
- Any non-empty text element that is a direct child of the encompassing element is supposed to be in simple Markdown (see here) and transformed into DocBook automatically.
- There are several additional elements that can be used on top of the normal DocBook and xtpxlib-xdoc elements. See “XprocRef specific DocBook extensions” on page 4.

3.3.1 XprocRef specific DocBook extensions

Element	Description

`<step>`
`<port>`
`<option>`
`<property>`
	Adds some special markup for the name of the step, port, option or (document) property (an HTML class with the same value).
	An empty `<step/>` element in a step description results in the step name itself.

`<step-error-ref code="…"/>`	Adds a reference to one of the step errors. The text of the reference is the error *code*.
`<step-ref name="…" version-id?=…"/>`	Adds a reference to a step. The name of the step must be including the namespace prefix (e.g. `name="p:identity"`). You can optionally refer to a step in another version by adding the `version-id` attribute.
`<category-ref idref="…"/>`	Adds a reference to a category. The text of the reference is the name of the category.
`<example-ref idref=…" step-name?=…"/>`	Adds a reference to an example. The text of the reference is the name of the example.
`<example-doc href="…"/>`	Inserts some example document. This is done as (unparsed) text, so make sure XML documents have no XML header!

3.3.1.1 Adding an XProc example with auto-execute

- It is possible to add XProc example pipelines that execute automatically. The result of this execution is available and shown as the result of the example pipeline.
- The XProc example pipeline must be self-running. That is, the input document must either be referenced (`@href`) on the `source` port or must be inlined. It will be presented for the example as a *separate* document. The reference or inlined document is removed from the pipeline in the example text.

```
<xproc-example href = xs:anyURI
               show-source? = xs:boolean
               show-pipeline? = xs:boolean
               show-result? = xs:boolean
               fixup-uris? = xs:boolean
               keep-from? = xs:string
               fixup-pipeline-input? = xs:boolean
               keep-namespace-prefixes? = List of xs:string
               keep-namespace-prefixes-source? = List of xs:string
               keep-namespace-prefixes-pipeline? = List of xs:string
               keep-namespace-prefixes-result? = List of xs:string
               output-is-text? >
  <source-header>?
  <pipeline-header>?
  <result-header>?
</xproc-example>
```

Attribute	#	Type	Description
`href`	1	`xs:anyURI`	The URI of the example pipeline.
`show-source`	?	`xs:boolean`	Default: `true` Whether to show the source document for the example pipeline.
`show-pipeline`	?	`xs:boolean`	Default: `true` Whether to show the source code of the example pipeline.
`show-result`	?	`xs:boolean`	Default: `true` Whether to show the result document of the example pipeline.
`fixup-uris`	?	`xs:boolean`	Default: `true` Whether to fixup any URIs (change the path to something bogus so source disk layout stays hidden).
`keep-from`	?	`xs:string`	When fixing up URIs, use the path including and after this string. If not set only the filename is used.
`fixup-pipeline-input`	?	`xs:boolean`	Default: `true` Whether to fixup the pipeline's input (that is, remove any references to it).
`keep-namespace-prefixes`	?	`List of xs:string`	A whitespace separated list of namespace-prefixes to keep, for all documents.
`keep-namespace-prefixes-source`	?	`List of xs:string`	A whitespace separated list of namespace-prefixes to keep for the source document.
`keep-namespace-prefixes-pipeline`	?	`List of xs:string`	A whitespace separated list of namespace-prefixes to keep for the pipeline document.
`keep-namespace-prefixes-result`	?	`List of xs:string`	A whitespace separated list of namespace-prefixes to keep for the result document.
`output-is-text`	?		Default: `false` Whether to treat the step's output as text and not as XML.

Child element	#	Description
`source-header`	?	DocBook text for the header above the source document. If absent, a default text is used. If empty, there will be no header.
`pipeline-header`	?	DocBook text for the header above the pipeline document. If absent, a default text is used. If empty, there will be no header.
`result-header`	?	DocBook text for the header above the result document. If absent, a default text is used. If empty, there will be no header.